**Normalisation:**

Three problems you can have with databases and why they are problems.

1. Too many “things” in one field, which makes it hard to search and sort.
2. Data repeated in many rows which means if something changes such as a customer address you need to change it in lots of places.
3. Tables that contain calculated values which take up processing power and unnecessary storage space if you have a really, really large database.

0NF – has all the problems

1NF – eliminate problem 1 (separate the data)

2NF – eliminate problem 1 and 2 (make multiple tables, add primary keys and link them)

3NF – eliminate all three problems. (eliminate calculated fields, put them in queries and reports)

**1NF:**

* There must be no repeated columns (fields) e.g. Contact person 1, Contact person 2, Contact person 3.
* there may only be one datum in any field - i.e. no "16kg" "3 minutes 56 seconds", "Large Size $4, Small $2.50", "16 Fred St, Melbourne, 3000".

2NF problems ONLY arise if you use a multi-field key (e.g. using firstname & familyname to uniquely identify people in a table).

2NF problems never even arise if each table has its own dedicated key field (e.g. ID, account number).

0NF

|  |  |  |  |
| --- | --- | --- | --- |
| Smith, Sally | 043123456 | French, English | Epilepsy |
| Jones, Alan | 043112345 | Greek, Architecture | Asthma |

1NF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Smith | Sally | 043123456 | French | Epilepsy |
| Smith | Sally | 043123456 | English | Epilepsy |
| Jones | Alan | 043112345 | Greek | Asthma |
| Jones | Alan | 043112345 | Architecture | Asthma |

And if Alan and Sally each had multiple ailments the number of rows for these two students would rapidly increase in 1NF.

**2NF requires:**

- 1NF has already been achieved.

- Any non-key field in a table is dependent on ALL of the fields used as the primary key.

Below is a table with the following fields. 1NF has already been achieved.

StudentID

SubjectID

Mark

SubjectName

**STUDENTID     SUBJECTID    Mark    SubjectName**

SMI0001       ENG          A+      English

SMI0001       MA           B       Maths

FRE0002       ENG          C       English

The table's key is STUDENTID and SUBJECTID (together) to uniquely identify each record in the table.

The (non-key) MARK field is dependent on both STUDENTID and SUBJECTID - i.e. to find out what a mark refers to, you need to know both the student and subject.

However, the (non-key) SUBJECTNAME field is dependent only the SUBJECTID - i.e. to find out what a subject name refers to, you only need the SUBJECTID. You don't need the STUDENTID.

So the (non-key) SUBJECTNAME field is dependent on **part of** the key (SUBJECTID) but not the **whole** key (STUDENTID+SUBJECTID).

So it fails 2NF.

To fix the problem, the table must be broken into two :

- MARKS\_TABLE with STUDENTID+SUBJECTID as its key. It also contains the MARK non-key field.

- SUBJECTS\_TABLE with SUBJECTID (primary key) and non-key SUBJECTNAME.

You then create a relationship between the MARKS table and the SUBJECTS table using their primary keys as the related fields.

Now, in the MARKS table, a Mark is dependent upon the entire key in its table (STUDENTID+SUBJECTID).

In the SUBJECTS table, a subject name is dependent on the entire key in its table (SUBJECTID).

This achieves 2NF.

**To achieve 3NF:**

- You must have already achieved 1NF and 2NF.

- No non-key field may be dependent on another non-key field.

Another way of saying it is that every non-key field in a table must give some information about the primary key rather than any other key in the table. Any field that does not contribute to the description of the primary key must be removed from the table.

For example... take a table. StudentID+SubjectID together are the primary key.

Honours is a Boolean field that is True if Mark is A or above, and False otherwise.

StudentID   SubjectID   Mark   Honours

ABC0001     S01         A      True

ABC0001     S02         A+     True

DEF0002     S01         B      False

The Honours field is dependent on the Mark field (i.e. to find the meaning of the Honours field, you need to refer to the Mark field) - but the Mark field is not the table's primary key. i.e. The Honours field describes the mark, not the student+subject.

So, a non-key field (Honours) is dependent on another non-key field (Mark). So it fails 3NF.

To fix it, do the same as we did before to achieve 2NF... break the offending field away into its own table with its own primary key (Mark) and non-key field (Honours) and relate the new table to the existing one using Mark as the link field.

This 3NF scenario looks VERY much like the 2NF before, doesn't it?

The only difference is that 2NF needed a non-key field to relate to the entire set of fields acting as the primary key.

In 3NF, it's actually simpler - a non-key field must not be dependent on another non-key field. In both cases, the fix is the same: table splitting. In my opinion, 3NF should be called 1.5NF because it seems to be more basic that 2NF, but we must all revere the word of Codd.

Always remember the normalisation oath:

The key (1NF), the whole key (2NF) and nothing but the key (3NF)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Customer | | |  | Phone | | | | ID | SN | FN | RecordNo | ID\_FK | Phone | | SAV01 | Savage | Laurie | 1 | SAV01 | 043123456 | | SMI05 | Smith | Sam | 2 | SAV01 | 0393062512 | |  |  |  |  | 3 | SMI05 | 0438111222 | |  |  |  |  | 4 | SMI05 | 0396541782 | |  |  |  |  | 5 | SMI05 | 0412345678 | | |  |  | | --- | --- | |  | | |  |  | |  |  | |  |  | |  |  | |

Customer table: primary key = ID; no repeating groups > 1NF; SN & FN are both dependent only on primary key > 2NF; neither SN or FN determine each other (ID is the only determinant) > 3NF

Phone table: first, why have the 'record number'? If it is an actual field then do you mean it to be a unique identifier? In other words, an ID field?

If this is the intention, then you are right: phone number is dependent on the meaningless ID and we have 3NF.

But, that is the same for any ID. Remember that the primary key is the smallest set of columns that uniquely identify a row. The reason we introduce an ID field is that in a table with many columns, the number of columns that make up the primary key could become quite unwieldy.

For your Customer table, for example, if you did not have ID, then the primary key would be SN+FN, and those two columns would have to be duplicated into any table you wished to link to the Customer table.

So, for the phone table - assuming it's going to be the 'destination' of a link, why bother with the RecordNo field at all. The primary key would be ID\_FK + Phone, and 1NF, 2NF and 3NF are achieved.

I know where you are coming from though - a few months ago I was trying to come up with some simple examples and found that reducing the table to two columns makes things very difficult for 2NF and 3NF. In effect the problems that 2NF and 3NF are trying to fix just disappear so 2NF and 3NF become pointless. The second thing that makes 2NF and 3NF 'disappear' is the use of ID fields because the problems that 2NF and 3NF are trying to fix only crop up when the primary key is made up of two or more columns.

So, examples that are trying to explain 2NF and 3NF need multi-column primary keys and at least one more column than just the primary key.

Or another way of looking at it:

*1NF - tidy the table(s) up horizontally  
ie Fields contain only a single value eg Name: Mr Barney Rubble changes to   
Title: Mr  
Fname: Barney   
Sname: Rubble  
  
Remove repeating Groups   
  
  
2NF - Tidy the table(s) up Vertically  
Eliminate duplication in records down the table by using a Primary Key that each record refers to.  
Continue to do this as often as required until duplication is removed from all tables- students should see the need for a new table if there is duplication with their own unique Primary Key.  
  
  
3NF - Make all fields in a table Mutually Exclusive  
ie, no field can rely on another.  eg table may contain "Quantity" and "Price", but it cannot contain "Total Cost" as Total cost = Quantity x Price.*